U.S. ARMY RDECOM

TECHNICAL REPORT RDMR-WD-11-28

DEVELOPMENT OF MODEL LIST CONVERSION APPLICATION

Milton A. Lamb
Weapons Development and Integration Directorate
Aviation and Missile Research, Development, and Engineering Center

September 2011

AMRDEC

DESTRUCTION NOTICE

FOR CLASSIFIED DOCUMENTS, FOLLOW THE PROCEDURES IN DoD 5200.22-M, INDUSTRIAL SECURITY MANUAL, SECTION II-19 OR DoD 5200.1-R, INFORMATION SECURITY PROGRAM REGULATION, CHAPTER IX. FOR UNCLASSIFIED, LIMITED DOCUMENTS, DESTROY BY ANY METHOD THAT WILL PREVENT DISCLOSURE OF CONTENTS OR RECONSTRUCTION OF THE DOCUMENT.

DISCLAIMER

THE FINDINGS IN THIS REPORT ARE NOT TO BE CONSTRUED AS AN OFFICIAL DEPARTMENT OF THE ARMY POSITION UNLESS SO DESIGNATED BY OTHER AUTHORIZED DOCUMENTS.

TRADE NAMES

USE OF TRADE NAMES OR MANUFACTURERS IN THIS REPORT DOES NOT CONSTITUTE AN OFFICIAL ENDORSEMENT OR APPROVAL OF THE USE OF SUCH COMMERCIAL HARDWARE OR SOFTWARE.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 2011	Final

4. TITLE AND SUBTITLE

Development of Model List Conversion Application

5. FUNDING NUMBERS

6. **AUTHOR**(S)

Milton A. Lamb

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Commander, U.S. Army Research, Development, and Engineering Command
ATTN: RDMR-WDG-C
Redstone Arsenal, AL 35898-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

TR-RDMR-WD-11-28

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited. Review completed by the AMRDEC Public Affairs Office; 22 November 2011; FN5590

12b. DISTRIBUTION CODE

A

13. ABSTRACT ***(Maximum 200 Words)***

ModelListConvert is an application to aid in the conversion of the Aviation and Missile Research, Development, and Engineering Center (AMRDEC) Weapons Development and Integration (WDI) Directorate Air Defense Artillery (ADA) trainer's model list file. The task was given to convert "modellist.bhv" from its original formatting to a spreadsheet format for dissemination. It became clear that a more efficient method was needed than to manually convert the file. An application, *ModelListConvert*, was created as the solution to this problem. This report outlines the process by which the application came to be and also serves as a manual for the application's use.

14. SUBJECT TERMS

Avenger, Trainer, Text File, Conversion, Application, Process, C#, Windows Forms

15. NUMBER OF PAGES

40

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

NSN 7540-**01**-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

TABLE OF CONTENTS

		Page
I.	INTRODUCTION	1
II.	TASK	2
III.	SOLUTION	2
IV.	PROCESS	2
	A. Original Goal	2
	B. User Interface	2
	C. Simple and Advanced Options	3
	D. Additional Capability—Conversion Back	4
	E. Computers Without Microsoft Office Installed	6
V.	CODE	8
VI.	CONCLUSIONS	8
	LIST OF ACRONYMS AND ABBREVIATIONS	9
	APPENDIX: SOURCE CODE	A-1

LIST OF ILLUSTRATIONS

Figure	Title	Page
1.	*ModelListConvert* Window	1
2.	Format of a ".bhv" File	1
3.	Default *ModelListConvert* Window	3
4.	Conversion from ".bhv" to ".xls"	4
5.	Format of the Excel Conversion	5
6.	Conversion from ".xls" to ".bhv"	6
7.	Conversion from ".bhv" to Tab Separated ".txt"	7
8.	Error Message for Correction	7
9.	Communication of *ModelListConvert*	8

I. INTRODUCTION

ModelListConvert is an application to aid in the conversion of the Aviation and Missile Research, Development, and Engineering Center (AMRDEC) Weapons Development and Integration (WDI) Directorate Air Defense Artillery (ADA) trainer's model list file. This file contains a detailed listing of all models available to the training device, including all information related to that model. This file is not the easiest file to edit or examine, so sometimes a conversion to a Microsoft Excel spreadsheet is desirable. This application allows this conversion to be performed automatically. The dialog for the application is shown in Figure 1.

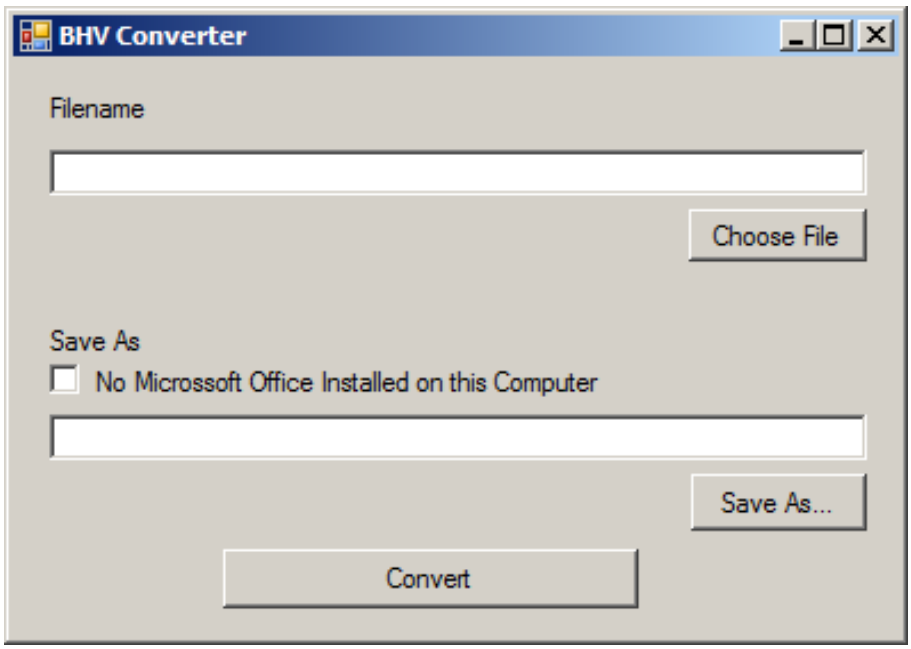


Figure 1. ModelListConvert Window

The model list file, which is named "modellist.bhv," is a text file formatted, as shown in Figure 2:

```
behavior {
    name = <string> "A319 Delta"
    classification = <string> fixedWing
    type = <string> "A319 Delta"
    appearance = <string> "Standard"
    id = <string> friendly
    description = <string> "France"
    modelFilename = <string> "A319_Delta.ive"
    dis {
        enum = <string> 1.2.71.57.1.5.*
        entityId = <int32> 32
        deadReckoning = <bool> true
        smoothPosition = <bool> false
        smoothOrientation = <bool> false
    }
}
```

Figure 2. Format of a ".bhv" File

One block, similar to this, exists for each entity type available to the trainer. An entity is a model—such as a plane, helicopter, or person—that is displayed during the run time of the trainer. Recent releases of the trainer and, more relevantly, the model list file have around 150 entities. Although it is a very useful way to store important information about entities for use programmatically, the file is not human friendly to read or edit.

II. TASK

The task was given to convert "modellist.bhv" from its original formatting to a spreadsheet format for dissemination. The conversion of all 150 entities (in the latest version as of January 2011) has traditionally been performed manually. This takes a significant amount of time that could be used in other more productive ways.

III. SOLUTION

It was clear that a faster and easier solution was needed. The first idea was the creation of a batch file to complete the process. Another solution involved the creation of a program to do the process. The batch file route was not chosen for several reasons. While a simple command with modifiers to run in the command line is tempting and quicker to write, it tends to alienate less computer-savvy users, and batch files do not tend to be as intuitive as a complete application, particularly one with a user interface. For this reason, creating a program was the chosen solution to the task.

IV. PROCESS

A. Original Goal

The first step was to create the substance of the program. This mainly consisted of writing the process for conversion from the ".bhv" format to an Excel-friendly format. The original goal was to convert the ".bhv" file to a comma delimited format. This process worked, but upon revision and testing, it was realized that there were still too many steps for it to be easily used and understood by most users. In addition, some of the fields use commas in their values. At a sacrifice for speed of the actual program but with improved user friendliness, it was decided to make the conversion directly to the native Excel format.

B. User Interface

The next step in the process was to give the program a Graphical User Interface (GUI). The format decided upon is shown in Figure 3.

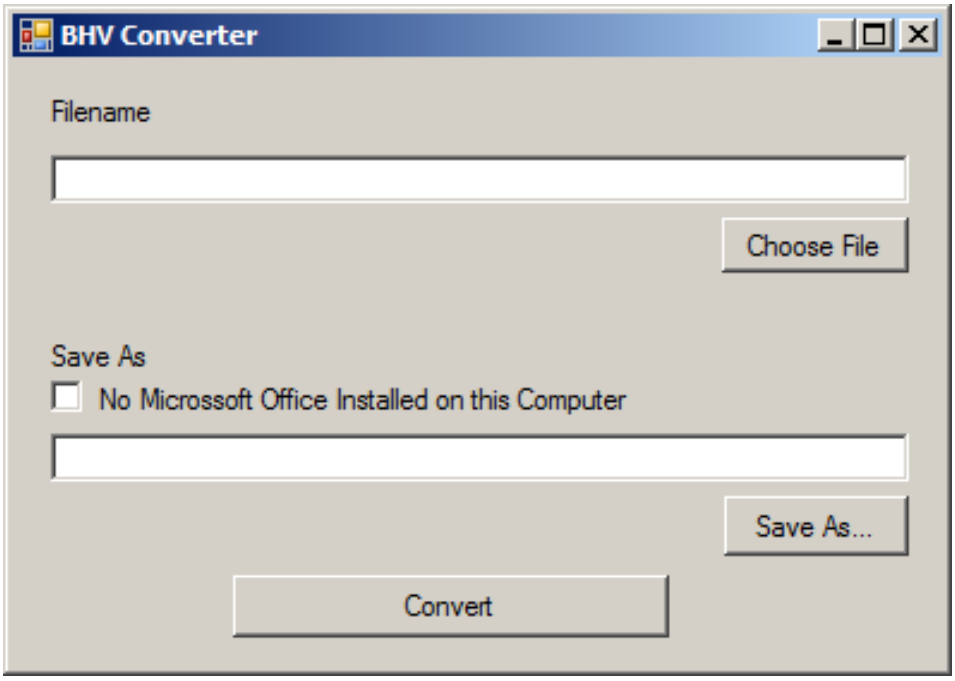


Figure 3. Default ModelListConvert Window

This format is simple and clean, and the idea was to make the conversion process fairly obvious. The user must either type the full path to the ".bhv" or click the "Choose File" button and choose the file to open. Then, the user must either type the full path of the filename desired or click the "Save As" button and select or input the filename as with a standard Windows program. When both of those options have been completed, the "Convert" button should be pressed to start the process. Due to the length of time before the conversion is completed, the cursor will change to an hourglass, and a dialog box with the text "Finished!" will appear when the process is over.

C. Simple and Advanced Options

The original goal of this process was to convert only some of the information to an Excel file. Only the non-redundant information was relevant for this process, but converting all of the information did not require much more code. Therefore, an option was added so that the user could choose to either output all or some of the information. Advanced (conversion of all of the information) is the default option. Choose Simple to only convert the basic data. Figure 4 shows the *ModelListConvert* dialog box with a standard full conversion from ".bhv" to ".xls" prepared.

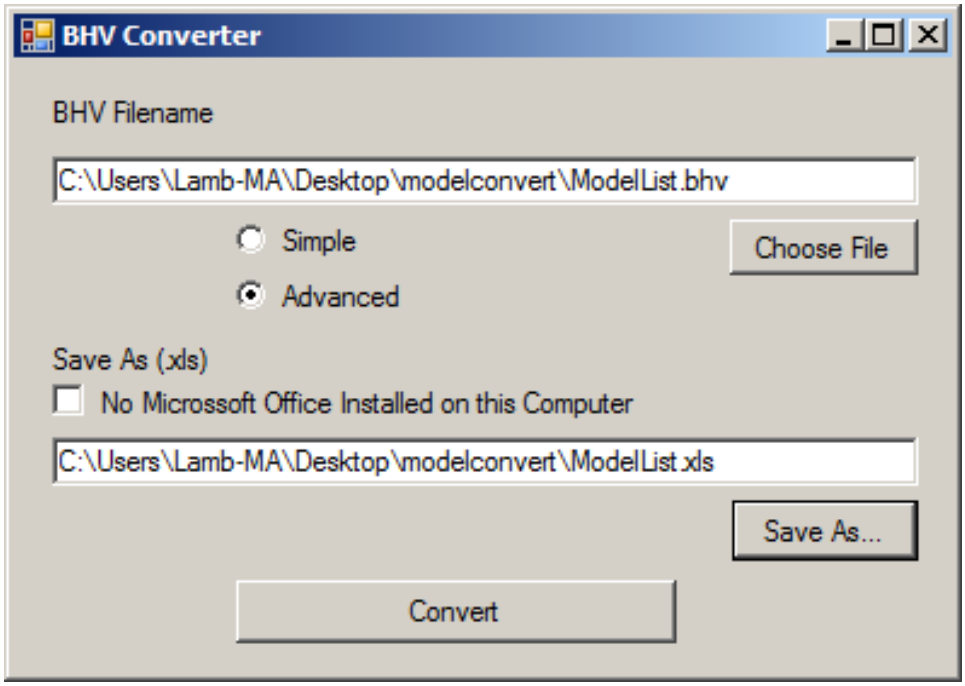
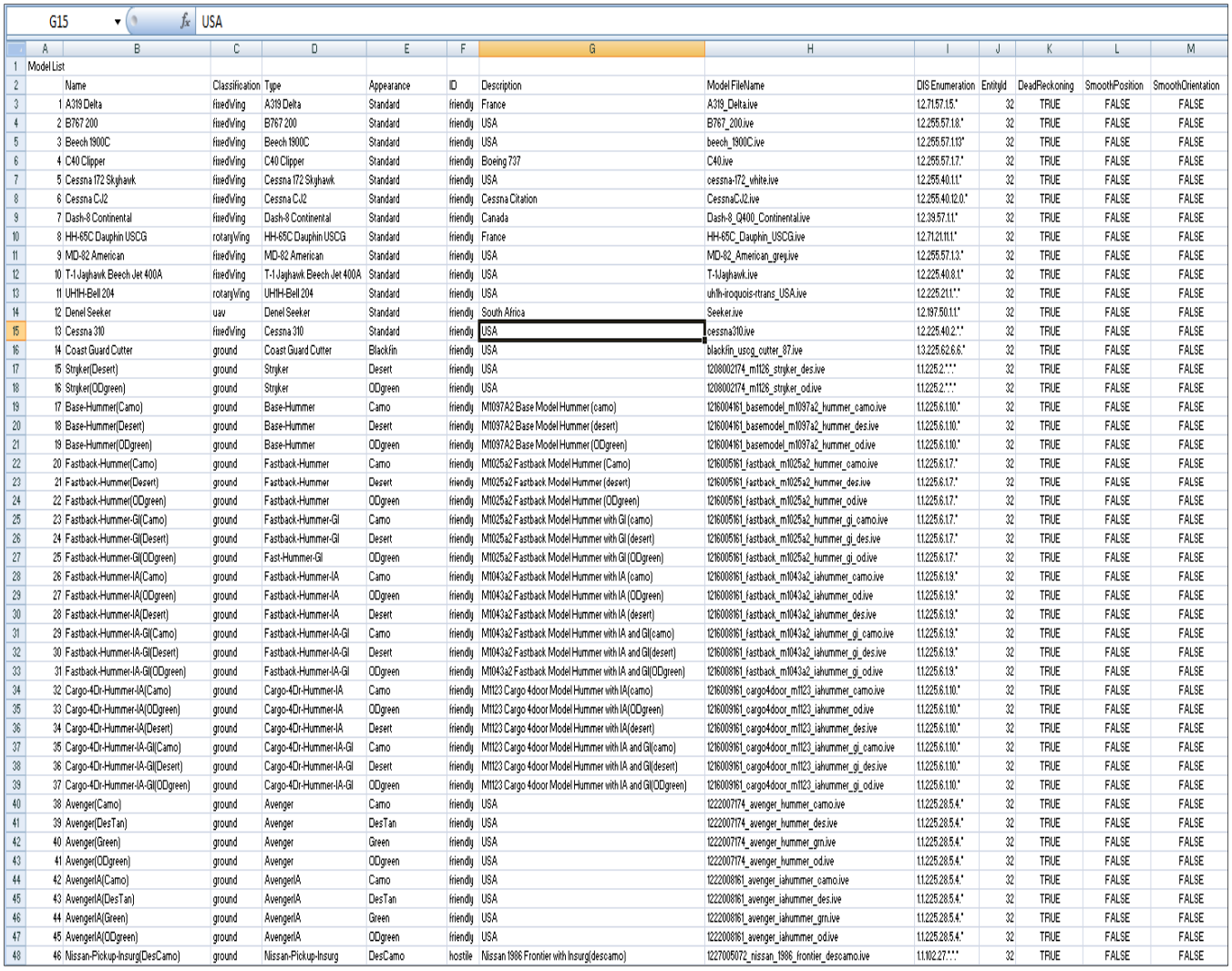


Figure 4. Conversion from ".bhv" to ".xls"

D. Additional Capability—Conversion Back

After the goal of converting the needed information to an Excel spreadsheet was achieved, it was also desirable for the information to be converted back to a ".bhv" formatted file. This decision was in large part due to the observation that it was much easier for the "modellist.bhv" file to be edited in spreadsheet format rather than ".bhv" format.

When the model list data is in the spreadsheet format, as shown in Figure 5, errors and typos can be seen much more easily, comparisons can be made faster, and rearrangement becomes more efficient and intuitive.



G15 USA

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Model List												
2		Name	Classification	Type	Appearance	ID	Description	Model FileName	DIS Enumeration	EntityId	DeadReckoning	SmoothPosition	SmoothOrientation
3	1	A319 Delta	fixedWing	A319 Delta	Standard	friendly	France	A319_Delta.ive	1.2.71.57.1.5.*	32	TRUE	FALSE	FALSE
4	2	B767 200	fixedWing	B767 200	Standard	friendly	USA	B767_200.ive	1.2.255.57.1.8.*	32	TRUE	FALSE	FALSE
5	3	Beech 1900C	fixedWing	Beech 1900C	Standard	friendly	USA	beech_1900C.ive	1.2.255.57.1.13*	32	TRUE	FALSE	FALSE
6	4	C40 Clipper	fixedWing	C40 Clipper	Standard	friendly	Boeing 737	C40.ive	1.2.255.57.1.7.*	32	TRUE	FALSE	FALSE
7	5	Cessna 172 Skyhawk	fixedWing	Cessna 172 Skyhawk	Standard	friendly	USA	cessna-172_white.ive	1.2.255.40.1.1.*	32	TRUE	FALSE	FALSE
8	6	Cessna CJ2	fixedWing	Cessna CJ2	Standard	friendly	Cessna Citation	CessnaCJ2.ive	1.2.255.40.12.0.*	32	TRUE	FALSE	FALSE
9	7	Dash-8 Continental	fixedWing	Dash-8 Continental	Standard	friendly	Canada	Dash-8_Q400_Continental.ive	1.2.39.57.1.1.*	32	TRUE	FALSE	FALSE
10	8	HH-65C Dauphin USCG	rotaryWing	HH-65C Dauphin USCG	Standard	friendly	France	HH-65C_Dauphin_USCG.ive	1.2.71.21.11.1.*	32	TRUE	FALSE	FALSE
11	9	MD-82 American	fixedWing	MD-82 American	Standard	friendly	USA	MD-82_American_grey.ive	1.2.255.57.1.3.*	32	TRUE	FALSE	FALSE
12	10	T-1 Jayhawk Beech Jet 400A	fixedWing	T-1 Jayhawk Beech Jet 400A	Standard	friendly	USA	T-1Jayhawk.ive	1.2.225.40.8.1.*	32	TRUE	FALSE	FALSE
13	11	UH1H-Bell 204	rotaryWing	UH1H-Bell 204	Standard	friendly	USA	uh1h-iroquois-rtrans_USA.ive	1.2.225.21.1.*.*	32	TRUE	FALSE	FALSE
14	12	Denel Seeker	uav	Denel Seeker	Standard	friendly	South Africa	Seeker.ive	1.2.197.50.1.1.*	32	TRUE	FALSE	FALSE
15	13	Cessna 310	fixedWing	Cessna 310	Standard	friendly	USA	cessna310.ive	1.2.225.40.2.*.*	32	TRUE	FALSE	FALSE
16	14	Coast Guard Cutter	ground	Coast Guard Cutter	Blackfin	friendly	USA	blackfin_uscg_cutter_87.ive	1.3.225.62.6.6.*	32	TRUE	FALSE	FALSE
17	15	Stryker(Desert)	ground	Stryker	Desert	friendly	USA	1208002174_m1126_stryker_des.ive	1.1.225.2.*.*.*	32	TRUE	FALSE	FALSE
18	16	Stryker(ODgreen)	ground	Stryker	ODgreen	friendly	USA	1208002174_m1126_stryker_od.ive	1.1.225.2.*.*.*	32	TRUE	FALSE	FALSE
19	17	Base-Hummer(Camo)	ground	Base-Hummer	Camo	friendly	M1097A2 Base Model Hummer (camo)	1216004161_basemodel_m1097a2_hummer_camo.ive	1.1.225.6.1.10.*	32	TRUE	FALSE	FALSE
20	18	Base-Hummer(Desert)	ground	Base-Hummer	Desert	friendly	M1097A2 Base Model Hummer (desert)	1216004161_basemodel_m1097a2_hummer_des.ive	1.1.225.6.1.10.*	32	TRUE	FALSE	FALSE
21	19	Base-Hummer(ODgreen)	ground	Base-Hummer	ODgreen	friendly	M1097A2 Base Model Hummer (ODgreen)	1216004161_basemodel_m1097a2_hummer_od.ive	1.1.225.6.1.10.*	32	TRUE	FALSE	FALSE
22	20	Fastback-Hummer(Camo)	ground	Fastback-Hummer	Camo	friendly	M1025a2 Fastback Model Hummer (Camo)	1216005161_fastback_m1025a2_hummer_camo.ive	1.1.225.6.1.7.*	32	TRUE	FALSE	FALSE
23	21	Fastback-Hummer(Desert)	ground	Fastback-Hummer	Desert	friendly	M1025a2 Fastback Model Hummer (desert)	1216005161_fastback_m1025a2_hummer_des.ive	1.1.225.6.1.7.*	32	TRUE	FALSE	FALSE
24	22	Fastback-Hummer(ODgreen)	ground	Fastback-Hummer	ODgreen	friendly	M1025a2 Fastback Model Hummer (ODgreen)	1216005161_fastback_m1025a2_hummer_od.ive	1.1.225.6.1.7.*	32	TRUE	FALSE	FALSE
25	23	Fastback-Hummer-GI(Camo)	ground	Fastback-Hummer-GI	Camo	friendly	M1025a2 Fastback Model Hummer with GI (camo)	1216005161_fastback_m1025a2_hummer_gi_camo.ive	1.1.225.6.1.7.*	32	TRUE	FALSE	FALSE
26	24	Fastback-Hummer-GI(Desert)	ground	Fastback-Hummer-GI	Desert	friendly	M1025a2 Fastback Model Hummer with GI (desert)	1216005161_fastback_m1025a2_hummer_gi_des.ive	1.1.225.6.1.7.*	32	TRUE	FALSE	FALSE
27	25	Fastback-Hummer-GI(ODgreen)	ground	Fast-Hummer-GI	ODgreen	friendly	M1025a2 Fastback Model Hummer with GI (ODgreen)	1216005161_fastback_m1025a2_hummer_gi_od.ive	1.1.225.6.1.7.*	32	TRUE	FALSE	FALSE
28	26	Fastback-Hummer-IA(Camo)	ground	Fastback-Hummer-IA	Camo	friendly	M1043a2 Fastback Model Hummer with IA (camo)	1216008161_fastback_m1043a2_iahummer_camo.ive	1.1.225.6.1.9.*	32	TRUE	FALSE	FALSE
29	27	Fastback-Hummer-IA(ODgreen)	ground	Fastback-Hummer-IA	ODgreen	friendly	M1043a2 Fastback Model Hummer with IA (ODgreen)	1216008161_fastback_m1043a2_iahummer_od.ive	1.1.225.6.1.9.*	32	TRUE	FALSE	FALSE
30	28	Fastback-Hummer-IA(Desert)	ground	Fastback-Hummer-IA	Desert	friendly	M1043a2 Fastback Model Hummer with IA (desert)	1216008161_fastback_m1043a2_iahummer_des.ive	1.1.225.6.1.9.*	32	TRUE	FALSE	FALSE
31	29	Fastback-Hummer-IA-GI(Camo)	ground	Fastback-Hummer-IA-GI	Camo	friendly	M1043a2 Fastback Model Hummer with IA and GI(camo)	1216008161_fastback_m1043a2_iahummer_gi_camo.ive	1.1.225.6.1.9.*	32	TRUE	FALSE	FALSE
32	30	Fastback-Hummer-IA-GI(Desert)	ground	Fastback-Hummer-IA-GI	Desert	friendly	M1043a2 Fastback Model Hummer with IA and GI(desert)	1216008161_fastback_m1043a2_iahummer_gi_des.ive	1.1.225.6.1.9.*	32	TRUE	FALSE	FALSE
33	31	Fastback-Hummer-IA-GI(ODgreen)	ground	Fastback-Hummer-IA-GI	ODgreen	friendly	M1043a2 Fastback Model Hummer with IA and GI(ODgreen)	1216008161_fastback_m1043a2_iahummer_gi_od.ive	1.1.225.6.1.9.*	32	TRUE	FALSE	FALSE
34	32	Cargo-4Dr-Hummer-IA(Camo)	ground	Cargo-4Dr-Hummer-IA	Camo	friendly	M1123 Cargo 4door Model Hummer with IA(camo)	1216009161_cargo4door_m1123_iahummer_camo.ive	1.1.225.6.1.10.*	32	TRUE	FALSE	FALSE
35	33	Cargo-4Dr-Hummer-IA(ODgreen)	ground	Cargo-4Dr-Hummer-IA	ODgreen	friendly	M1123 Cargo 4door Model Hummer with IA(ODgreen)	1216009161_cargo4door_m1123_iahummer_od.ive	1.1.225.6.1.10.*	32	TRUE	FALSE	FALSE
36	34	Cargo-4Dr-Hummer-IA(Desert)	ground	Cargo-4Dr-Hummer-IA	Desert	friendly	M1123 Cargo 4door Model Hummer with IA(desert)	1216009161_cargo4door_m1123_iahummer_des.ive	1.1.225.6.1.10.*	32	TRUE	FALSE	FALSE
37	35	Cargo-4Dr-Hummer-IA-GI(Camo)	ground	Cargo-4Dr-Hummer-IA-GI	Camo	friendly	M1123 Cargo 4door Model Hummer with IA and GI(camo)	1216009161_cargo4door_m1123_iahummer_gi_camo.ive	1.1.225.6.1.10.*	32	TRUE	FALSE	FALSE
38	36	Cargo-4Dr-Hummer-IA-GI(Desert)	ground	Cargo-4Dr-Hummer-IA-GI	Desert	friendly	M1123 Cargo 4door Model Hummer with IA and GI(desert)	1216009161_cargo4door_m1123_iahummer_gi_des.ive	1.1.225.6.1.10.*	32	TRUE	FALSE	FALSE
39	37	Cargo-4Dr-Hummer-IA-GI(ODgreen)	ground	Cargo-4Dr-Hummer-IA-GI	ODgreen	friendly	M1123 Cargo 4door Model Hummer with IA and GI(ODgreen)	1216009161_cargo4door_m1123_iahummer_gi_od.ive	1.1.225.6.1.10.*	32	TRUE	FALSE	FALSE
40	38	Avenger(Camo)	ground	Avenger	Camo	friendly	USA	1222007174_avenger_hummer_camo.ive	1.1.225.28.5.4.*	32	TRUE	FALSE	FALSE
41	39	Avenger(DesTan)	ground	Avenger	DesTan	friendly	USA	1222007174_avenger_hummer_des.ive	1.1.225.28.5.4.*	32	TRUE	FALSE	FALSE
42	40	Avenger(Green)	ground	Avenger	Green	friendly	USA	1222007174_avenger_hummer_grn.ive	1.1.225.28.5.4.*	32	TRUE	FALSE	FALSE
43	41	Avenger(ODgreen)	ground	Avenger	ODgreen	friendly	USA	1222007174_avenger_hummer_od.ive	1.1.225.28.5.4.*	32	TRUE	FALSE	FALSE
44	42	AvengerIA(Camo)	ground	AvengerIA	Camo	friendly	USA	1222008161_avenger_iahummer_camo.ive	1.1.225.28.5.4.*	32	TRUE	FALSE	FALSE
45	43	AvengerIA(DesTan)	ground	AvengerIA	DesTan	friendly	USA	1222008161_avenger_iahummer_des.ive	1.1.225.28.5.4.*	32	TRUE	FALSE	FALSE
46	44	AvengerIA(Green)	ground	AvengerIA	Green	friendly	USA	1222008161_avenger_iahummer_grn.ive	1.1.225.28.5.4.*	32	TRUE	FALSE	FALSE
47	45	AvengerIA(ODgreen)	ground	AvengerIA	ODgreen	friendly	USA	1222008161_avenger_iahummer_od.ive	1.1.225.28.5.4.*	32	TRUE	FALSE	FALSE
48	46	Nissan-Pickup-Insurg(DesCamo)	ground	Nissan-Pickup-Insurg	DesCamo	hostile	Nissan 1986 Frontier with Insurg(descamo)	1227005072_nissan_1986_frontier_descamo.ive	1.1.102.27.*.*.*	32	TRUE	FALSE	FALSE

Figure 5. Format of the Excel Conversion

The next step in the development process was to decide how to implement the additional capability into the original program. Consideration was made to the idea of creating a separate program, but it was thought that having two programs would be more difficult to deal with than just one. Therefore, the decision was made to add the new code to the same application. The last thing that needed to be decided was where the conversion should take place—in a new window, a new tab, or on the same window with no tabs. For simplicity, it was decided to keep it on the same window and to not include tabs. In this manner, either an ".xls," ".bhv," or ".txt" file can be put into the first text box, and code was added to recognize which type was entered such that the corresponding extension was the only available option under "Save As." An example is that if a ".bhv" file was chosen in the first text box, then only an Excel spreadsheet or ".txt" file would be allowed to be chosen under the "Save As" option. The opposite is also true. For this reason, the original file must be chosen first. A dialog box will appear explaining the proper method if this process is not followed. Alternatively, the typing of the paths can occur in either order. Figure 6, in comparison to Figure 4, shows how the program can recognize differing inputs and changes, accordingly.

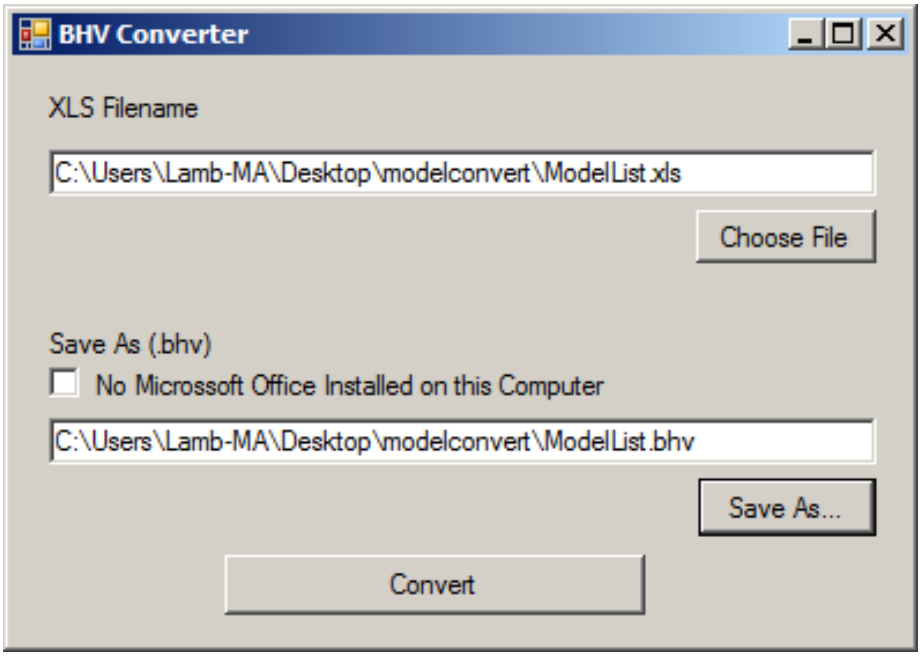


Figure 6. Conversion from ".xls" to ".bhv"

E. Computers Without Microsoft Office Installed

After the code was thought to be complete, an error was noticed on computers without Microsoft Office installed. The error stems from the fact that Microsoft Office libraries are being used in the conversion process. The libraries are installed during the Microsoft Office installation process. There were three options available to fix this. The simplest, of course, was to just require that the program not be used on a computer without Microsoft Office installed. This was not desirable, as this would severely limit the usage of the program. Computers with a different spreadsheet program installed would be unable to perform the process, and the process could only be used as long as the Microsoft libraries did not change (for example, old versions of Microsoft libraries and one created in the future may be incompatible). So the other choices were to either find a way to include the needed libraries or create a separate process that, while less straightforward, is more reliable.

To include the Microsoft libraries with the program would call for some sort of installation process. However, this is undesirable since the architecture for the program has been simplicity whenever possible. On the other hand, creating a separate process allows the program to be kept as a standalone program that is able to be run by simply having it on a machine.

A checkbox was added, to be checked in the case of a computer not having Excel and Microsoft Office installed. When this is checked and ".bhv" is chosen to be converted, a text file output becomes the only option under the "Save As" dialog box. This text file will be tab separated in spreadsheet format and is shown in Figure 7.

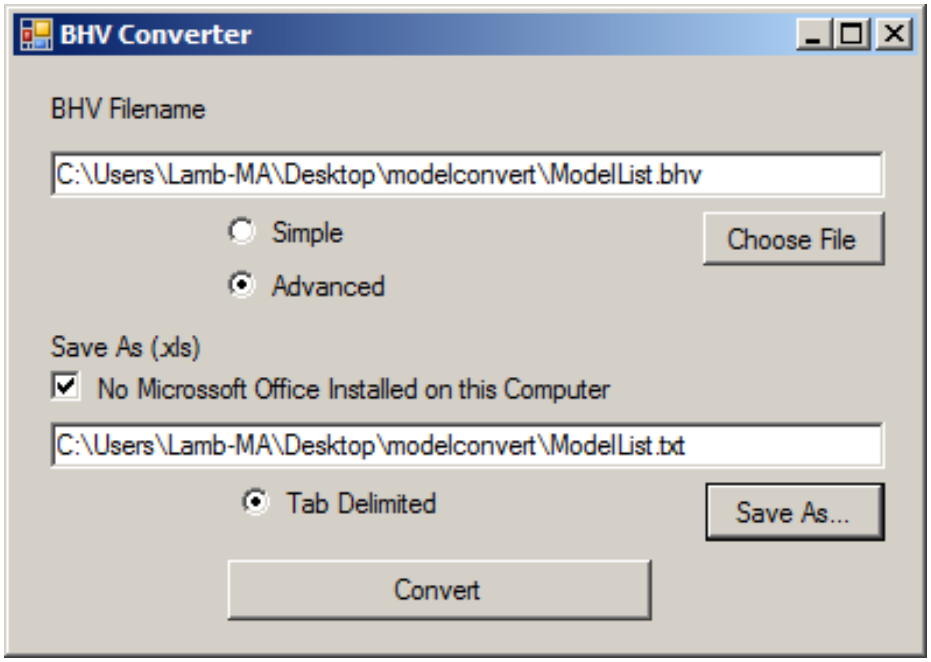


Figure 7. Conversion from ".bhv" to Tab Separated ".txt"

Figure 8 shows that the error message occurs when the "Convert" button is pressed, the "No Microsoft Office" checkbox is checked, and the Excel file format was chosen in the "Save As" text box. This will usually occur when the "Save As" button is pressed before the checkbox has been checked. The display of this error should not cause any errors in the execution.

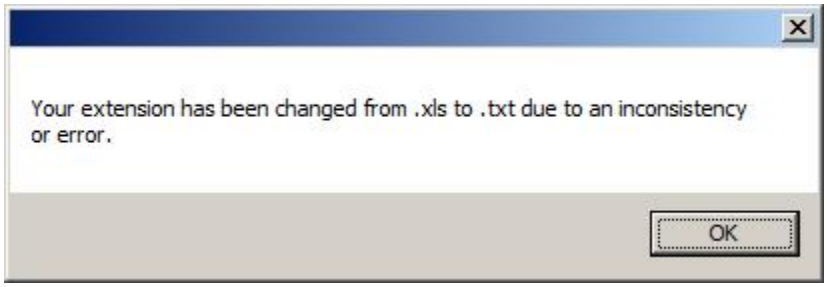


Figure 8. Error Message for Correction

Similarly, an error message will be displayed in the reverse process. This will most often occur in the event the "Save As" button is pressed while the checkbox is checked and then, the checkbox is unchecked after a "Save File" is chosen. A similar error message will be displayed in the event of the "Save As" text box having a ".txt" file in it when the box is unchecked. In the reverse process of ".txt" to ".bhv," the status of the checkbox is irrelevant as both processes will be the same.

V. CODE

The architecture of the program, including the user interface, is very simple and linear, as illustrated in Figure 9. It consists of the main file "Program.cs," the general user interface auto-generated code (not included in the Appendix), "BHV Convert.cs," and "Convert.cs." The entire program was written in C# using Visual Studio 2010.

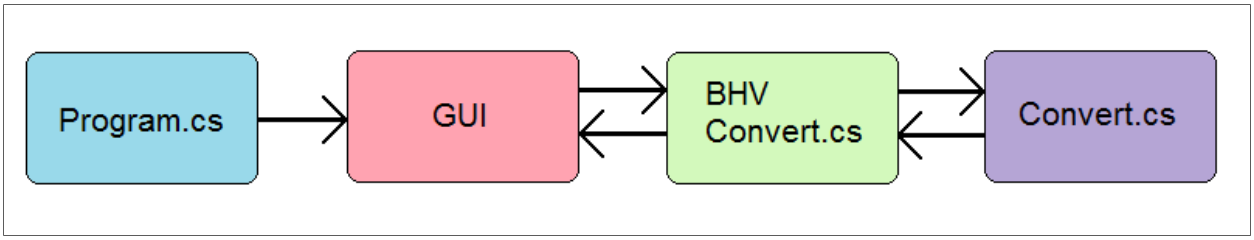


Figure 9. Communication of ModelListConvert

"Program.cs" contains the code for starting the application. While not necessary, it was included for good programming practice. "Program.cs" starts the GUI, which communicates with "BHV Convert.cs." "BHV Convert.cs" includes all of the commands for each control on the GUI (such as the buttons and checkboxes). The "Convert" button on the GUI calls the "btn_Convert_Click" function, which calls "Convert.cs." "Convert.cs" then sends information back to "BHV Convert.cs," and the information is displayed on the GUI. "Convert.cs" contains all of the code to actually convert the file, and all of the different options available are contained within it.

VI. CONCLUSION

This report shows how a program evolves as the goals it needs to achieve become clearer and change. It also shows that simplicity can be very important and explains the process for the specific creation of the file converter, *ModelListConvert*.

This project began with a simple goal to "Convert the modellist.bhv file into something more accessible in spreadsheet form." It ended up being, "Create a program that can convert a file in ".bhv" format to either an Excel spreadsheet or a tab separated text file that can be opened in a spreadsheet reading program, and then convert it back again." Even though the program gained more capability and complexity, careful coding enabled the project's user interface to be kept quite simple. More concretely, however, this report provides documentation for the *ModelListConvert* program and explanation of its use.

LIST OF ACRONYMS AND ABBREVIATIONS

ADA	Air Defense Artillery
AMRDEC	Aviation and Missile Research, Development, and Engineering Center
GUI	Graphical User Interface
WDI	Weapons Development and Integration

APPENDIX
SOURCE CODE

I. PROGRAM.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace ModelListConvert
{
   static class Program
   {
      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new BHV_Converter());
      }
   }
}
```

II. BHV CONVERT.CS

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace ModelListConvert
{
   public partial class BHV_Converter : Form
   {
      string m_ReadFileName;
      string m_SaveFileName;
      bool m_Cansave = false;

      Convert convert = new Convert();

      public BHV_Converter()
      {
         InitializeComponent();
```

```
    }

    private void openFileDialog1_FileOk(object sender, CancelEventArgs e)
    {

    }

    private void btn_ChooseFile_Click(object sender, EventArgs e)
    {
        //creates a filter so that the user can only open a bhv, txt file, or excel
        OpenFileDialog openFile = new OpenFileDialog();
        openFile.Filter = "Supported Files Types|*.bhv;*.txt;*.xls|(*.bhv,*.txt)|*.bhv;*.txt|bhv
files (*.bhv)| *.bhv|txt files" +
            "(*.txt)|*.txt|xls files(*.xls)|*.xls|All files (*.*)|*.*";
        openFile.ShowDialog();
        txt_filename.Text = openFile.FileName;
        m_ReadFileName = txt_filename.Text;

        //changes the labels to accurately represent the covnersion process
        if (m_ReadFileName.EndsWith(".bhv") && cb_NOMO.Checked == false)
        {
            //this adds the radio buttons that are relevant to this way of saving the file
            rad_Simple.Visible = true;
            rad_Adv.Visible = true;

            label1.Text = "BHV Filename";
            label2.Text = "Save As (.xls)";
        }
        else if (m_ReadFileName.EndsWith(".bhv") && cb_NOMO.Checked == true)
        {
            rad_Simple.Visible = true;
            rad_Adv.Visible = true;

            label1.Text = "BHV Filename";
            label2.Text = "Save As (.txt, tab separated)";
        }
        else if (m_ReadFileName.EndsWith(".txt"))
        {
            rad_Simple.Visible = false;
            rad_Adv.Visible = false;

            label1.Text = "TEXT (tab separated) Filename";
            label2.Text = "Save As (.bhv)";
        }

        else
```

```
        {
            rad_Simple.Visible = false;
            rad_Adv.Visible = false;

            label1.Text = "XLS Filename";
            label2.Text = "Save As (.bhv)";
        }
        m_Cansave = true;
    }

    private void btn_SaveFile_Click(object sender, EventArgs e)
    {
        SaveFileDialog savefile = new SaveFileDialog();

        if (File.Exists(txt_filename.Text) == false)
        {
            MessageBox.Show("Open File does not exist");
        }
        else if (m_Cansave == true && m_ReadFileName != null)
        {
            //if the file is text or .bhv file then it only lets you save in an .xls format
            if (/*m_ReadFileName.EndsWith(".txt") ||*/ m_ReadFileName.EndsWith(".bhv"))
            {
                if (cb_NOMO.Checked == true)
                {
                    savefile.Filter = "txt files (*.txt)|*.txt";
                    savefile.ShowDialog();
                    txt_SaveAs.Text = savefile.FileName;
                    m_SaveFileName = savefile.FileName;
                }
                else
                {
                    savefile.Filter = "xls files (*.xls)|*.xls";
                    savefile.ShowDialog();
                    txt_SaveAs.Text = savefile.FileName;
                    m_SaveFileName = savefile.FileName;
                }
            }

            //if the file is an .xls file then it only lets you save in a text or .bhv format
            else if (m_ReadFileName.EndsWith(".xls") || m_ReadFileName.EndsWith(".xlsx") ||
m_ReadFileName.EndsWith(".txt"))
            {
                savefile.Filter = "bhv files (*.bhv)|*.bhv";
                savefile.ShowDialog();
                txt_SaveAs.Text = savefile.FileName;
```

```
            m_SaveFileName = savefile.FileName;
        }
    }
    else
    {
        MessageBox.Show("Please choose a file to open first.");
    }
}

private void btn_Convert_Click(object sender, EventArgs e)
{

    bool simple = rad_Simple.Checked;
    bool tab = rad_Tab.Checked;
    bool finished = true;
    string to = null;
    string from = null;

    m_ReadFileName = txt_filename.Text;
    m_SaveFileName = txt_SaveAs.Text;

    //sets the boolean for whether the end file will be a spreadsheet or not.
    if(m_ReadFileName == null || m_SaveFileName == null)
    {
        MessageBox.Show("Your file paths are not correct.");
        finished = false;
    }
    else if (/*m_ReadFileName.EndsWith(".txt") ||*/ m_ReadFileName.EndsWith(".bhv")
&& cb_NOMO.Checked && m_SaveFileName.EndsWith(".txt"))
    {
        from = "bhv";
        to = "txt";
    }
    else if (m_ReadFileName.EndsWith(".bhv") && m_SaveFileName.EndsWith(".xls") &&
cb_NOMO.Checked == false)
    {
        from = "bhv";
        to = "xls";
    }
    else if (m_ReadFileName.EndsWith(".txt") && cb_NOMO.Checked)
    {
        from = "txt";
        to = "bhv";
    }
    else if (m_ReadFileName.EndsWith(".xls") || m_ReadFileName.EndsWith(".xlsx"))
    {
```

```
            from = "xls";
            to = "bhv";
        }

        else if (m_ReadFileName.EndsWith(".bhv") && (m_SaveFileName.EndsWith(".xls") ||
m_SaveFileName.EndsWith(".xlsx")) && cb_NOMO.Checked)
        {
            MessageBox.Show("Your extension has been changed from .xls to .txt due to an
inconsistency or error.");
            string correctedFN = txt_SaveAs.Text;
            int periodpos = 0;
            for (int i = 0; i < correctedFN.Length; i++)
            {
                if (correctedFN[i] == '.')
                {
                    periodpos = i;
                }
            }
            m_SaveFileName = correctedFN.Substring(0, periodpos) + ".txt";
            from = "bhv";
            to = "txt";
        }
        else if (m_ReadFileName.EndsWith(".bhv") && m_SaveFileName.EndsWith(".txt") &&
cb_NOMO.Checked == false)
        {
            MessageBox.Show("Your extension has been changed from .txt to .xls due to an
inconsistency or error.");
            string correctedFN = txt_SaveAs.Text;
            int periodpos = 0;
            for (int i = 0; i < correctedFN.Length; i++)
            {
                if (correctedFN[i] == '.')
                {
                    periodpos = i;
                }
            }
            m_SaveFileName = correctedFN.Substring(0, periodpos) + ".xls";
            from = "bhv";
            to = "xls";

        }
        else
        {
            MessageBox.Show("Your file paths are not correct.");
            finished = false;
        }
```

```
        //checks for if the the member filenames have been set via the choose file
        //and save as buttons and if not it tries to create the names another way

        if (m_SaveFileName == "" || m_SaveFileName == null)
        {
            string correctedFN = txt_filename.Text;
            int periodpos = 0;
            for (int i = 0; i < correctedFN.Length; i++)
            {
                if (correctedFN[i] == '.')
                {
                    periodpos = i;
                }
            }

            if (cb_NOMO.Checked) { m_SaveFileName = correctedFN.Substring(0, periodpos) +
".txt"; }
            else{m_SaveFileName = correctedFN.Substring(0, periodpos) + ".xls";}
        }
        //runs the conversion
        Cursor temp = Cursor.Current;
        Cursor.Current = Cursors.WaitCursor;
        this.Text = "BHV Converter  Working...";
        convert.convert(m_ReadFileName, m_SaveFileName, from, to, simple, tab);
        //this appears when the conversion is finished
        Cursor.Current = temp;
        this.Text = "BHV Converter";
        if (finished) { MessageBox.Show("Finished"); }
    }

    private void txt_filename_TextChanged(object sender, EventArgs e)
    {
        m_Cansave = true;
        m_ReadFileName = txt_filename.Text;
    }

    private void checkBox1_CheckedChanged(object sender, EventArgs e)
    {
        if (cb_NOMO.Checked == true)
        {
            rad_Tab.Checked = true;
            rad_Tab.Visible = true;
        }
        else
        {
            rad_Tab.Checked = false;
```

```
            rad_Tab.Visible = false;
        }
    }
  }
}
```

III. CONVERT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using Excel = Microsoft.Office.Interop.Excel;

namespace ModelListConvert
{
  class Convert
  {
    int m_Quote1;
    int m_Quote2;
    int m_Chevron;

    public Convert()
    {

    }

    public void convert(string ReadFileName, string SaveFileName, string from, string to, bool
simple, bool tab)
    {

      string delim;
      if (from  == "bhv" && to == "txt")
      {
        delim  = "\t";
        if (simple)
        {
          convertSimpleDelim(ReadFileName, SaveFileName, delim);
        }
        else
        {
          convertAdvDelim(ReadFileName, SaveFileName, delim);
        }
      }
      else if (from == "bhv" && to == "xls")
```

```
    {
        if (simple)
        {
            convSimpleXls(ReadFileName, SaveFileName);
        }
        else
        {
            convAdvXls(ReadFileName, SaveFileName);
        }
    }
    else if (from == "xls" && to == "bhv")
    {
        convertBackxls(ReadFileName, SaveFileName);
    }
    else if (from == "txt" && to == "bhv")
    {
        convertBacktxt(ReadFileName, SaveFileName);
    }
}

//set the position of the two quotes of a given string to m_Quote1 & 2
void findQuotes(string input)
{
    bool found = true;
    for (int i = 0; i < input.Length; i++)
    {
        if (found && input[i] == '\"')
        {
            m_Quote1 = i;
            found = false;
        }
        else if (!found && input[i] == '\"')
        {
            m_Quote2 = i;
        }

    }
}

//sets the position of the greater than ('>') in a given string to m_Chevron
void findChevron(string input)
{
    for (int i = 0; i < input.Length; i++)
    {
        if (input[i] == '>')
        {
```

```
            m_Chevron = i;
        }


    }
}


//converts from .xls to .bhv or .txt
void convertBackxls(string ReadFileName, string SaveFileName)
{
    string readline;
    string d = "\t";
    string[] writeline = new String[12];
    System.IO.TextWriter tw = new System.IO.StreamWriter(SaveFileName);

    Excel.Application xlApp;
    Excel.Workbook xlWorkBook;
    Excel.Worksheet xlWorkSheet;
    Excel.Range range;

    xlApp = new Excel.Application();
    xlWorkBook = xlApp.Workbooks.Open(ReadFileName, 0, true, 5, "", "", true,
Microsoft.Office.Interop.Excel.XlPlatform.xlWindows, d, false, false, 0, true, 1, 0);
    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);
    range = xlWorkSheet.UsedRange;

    tw.WriteLine("behaviorModels {");

    //reads through the excel sheet and prints out the contents in the correct format
    for(int i = 3; i<=range.Rows.Count; i++)
    {
        readline = "";
        for (int j = 2; j <= range.Columns.Count; j++)
        {
            object cellvalue = (range.Cells[i, j] as Excel.Range).Value2;
            if (cellvalue != null)
            {
                readline = readline + d + (range.Cells[i, j] as Excel.Range).Value2.ToString();
            }

        }


        Regex delim = new Regex(d);
        String[] strarray = delim.Split(readline);
```

```
//checks to see if it is converting from a basic spreadsheet or advanced
if (strarray.Length == 7)
{
   tw.WriteLine("\tbehavior {");
   tw.WriteLine("\t\tname = <string> \"" + strarray[1] + "\"");
   tw.WriteLine("\t\tclassification = <string> " + strarray[2]);
   tw.WriteLine("\t\ttype = <string> \"" + strarray[1] + "\"");
   tw.WriteLine("\t\tappearance = <string> \"Standard\"");
   tw.WriteLine("\t\tid = <string> " + strarray[3]);
   tw.WriteLine("\t\tdescription = <string> \"" + strarray[4] + "\"");
   tw.WriteLine("\t\tmodelFilename = <string> \"" + strarray[5] + "\"");
   tw.WriteLine("\t\tdis {");
   tw.WriteLine("\t\t\tenum = <string> " + strarray[6]);
   tw.WriteLine("\t\t\tentityId = <int32> 32");
   tw.WriteLine("\t\t\tdeadReckoning = <bool> true");
   tw.WriteLine("\t\t\tsmoothPosition = <bool> false");
   tw.WriteLine("\t\t\tsmoothOrientation = <bool> false");
   tw.WriteLine("\t\t}");
   tw.WriteLine("\t}");
}
else if (strarray.Length >= 13)
{
   tw.WriteLine("\tbehavior {");
   tw.WriteLine("\t\tname = <string> \"" + strarray[1] + "\"");
   tw.WriteLine("\t\tclassification = <string> " + strarray[2]);
   tw.WriteLine("\t\ttype = <string> \"" + strarray[3] + "\"");
   tw.WriteLine("\t\tappearance = <string> \"" + strarray[4] + "\"");
   tw.WriteLine("\t\tid = <string> " + strarray[5]);
   tw.WriteLine("\t\tdescription = <string> \"" + strarray[6] + "\"");
   tw.WriteLine("\t\tmodelFilename = <string> \"" + strarray[7] + "\"");
   tw.WriteLine("\t\tdis {");
   tw.WriteLine("\t\t\tenum = <string> " + strarray[8]);
   tw.WriteLine("\t\t\tentityId = <int32> " + strarray[9]);
   tw.WriteLine("\t\t\tdeadReckoning = <bool> " + strarray[10].ToLower());
   tw.WriteLine("\t\t\tsmoothPosition = <bool> " + strarray[11].ToLower());
   tw.WriteLine("\t\t\tsmoothOrientation = <bool> " + strarray[12].ToLower());
   tw.WriteLine("\t\t}");


   if (strarray.Length == 17)
   {
      tw.WriteLine("\t\taccessories = <string> \"" + strarray[14] + "\"");
      tw.WriteLine("\t\tidleAnimations = <string> \"" + strarray[15] + "\"");
      tw.WriteLine("\t\tmoveAnimations = <string> \"" + strarray[16] + "\"");
      tw.WriteLine("\t\tmachineGunRoundsToDestroy = <int32> " + strarray[13]);
   }
```

```
            else if (strarray.Length == 14)
            {
                tw.WriteLine("\t\tmachineGunRoundsToDestroy = <int32> " + strarray[13]);
            }


            tw.WriteLine("\t}");
        }

    }
    tw.WriteLine("}");

    xlWorkBook.Close(true, null, null);
    xlApp.Quit();
    tw.Close();
}

void convertBacktxt(string ReadFileName, string SaveFileName)
{
    string readline;
    string d = "\t";
    string[] writeline = new String[12];
    System.IO.TextWriter tw = new System.IO.StreamWriter(SaveFileName);
    System.IO.TextReader tr = new System.IO.StreamReader(ReadFileName);


    tw.WriteLine("behaviorModels {");
    tr.ReadLine();
    tr.ReadLine();
    readline = tr.ReadLine();
    String[] strarray;
    //reads through the excel sheet and prints out the contents in the correct format
    while (readline != null)
    {

        strarray = null;
        Regex delim = new Regex(d);
        strarray = delim.Split(readline.Trim());
        //checks to see if it is converting from a basic spreadsheet or advanced
        if (strarray.Length == 7)
        {
            tw.WriteLine("\tbehavior {");
            tw.WriteLine("\t\tname = <string> \"" + strarray[1] + "\"");
            tw.WriteLine("\t\tclassification = <string> " + strarray[2]);
            tw.WriteLine("\t\ttype = <string> \"" + strarray[1] + "\"");
            tw.WriteLine("\t\tappearance = <string> \"Standard\"");
```

```
    tw.WriteLine("\t\tid = <string> " + strarray[3]);
    tw.WriteLine("\t\tdescription = <string> \"" + strarray[4] + "\"");
    tw.WriteLine("\t\tmodelFilename = <string> \"" + strarray[5] + "\"");
    tw.WriteLine("\t\tdis {");
    tw.WriteLine("\t\t\tenum = <string> " + strarray[6]);
    tw.WriteLine("\t\t\tentityId = <int32> 32");
    tw.WriteLine("\t\t\tdeadReckoning = <bool> true");
    tw.WriteLine("\t\t\tsmoothPosition = <bool> false");
    tw.WriteLine("\t\t\tsmoothOrientation = <bool> false");
    tw.WriteLine("\t\t}");
    tw.WriteLine("\t}");
}
else if (strarray.Length >= 13)
{
    tw.WriteLine("\tbehavior {");
    tw.WriteLine("\t\tname = <string> \"" + strarray[1] + "\"");
    tw.WriteLine("\t\tclassification = <string> " + strarray[2]);
    tw.WriteLine("\t\ttype = <string> \"" + strarray[3] + "\"");
    tw.WriteLine("\t\tappearance = <string> \"" + strarray[4] + "\"");
    tw.WriteLine("\t\tid = <string> " + strarray[5]);
    tw.WriteLine("\t\tdescription = <string> \"" + strarray[6] + "\"");
    tw.WriteLine("\t\tmodelFilename = <string> \"" + strarray[7] + "\"");
    tw.WriteLine("\t\tdis {");
    tw.WriteLine("\t\t\tenum = <string> " + strarray[8]);
    tw.WriteLine("\t\t\tentityId = <int32> " + strarray[9]);
    tw.WriteLine("\t\t\tdeadReckoning = <bool> " + strarray[10].ToLower());
    tw.WriteLine("\t\t\tsmoothPosition = <bool> " + strarray[11].ToLower());
    tw.WriteLine("\t\t\tsmoothOrientation = <bool> " + strarray[12].ToLower());
    tw.WriteLine("\t\t}");


    if (strarray.Length == 17)
    {
        tw.WriteLine("\t\taccessories = <string> \"" + strarray[14] + "\"");
        tw.WriteLine("\t\tidleAnimations = <string> \"" + strarray[15] + "\"");
        tw.WriteLine("\t\tmoveAnimations = <string> \"" + strarray[16] + "\"");
        tw.WriteLine("\t\tmachineGunRoundsToDestroy = <int32> " + strarray[13]);
    }
    else if (strarray.Length == 14)
    {
        tw.WriteLine("\t\tmachineGunRoundsToDestroy = <int32> " + strarray[13]);
    }


    tw.WriteLine("\t}");
}
```

```
        readline = tr.ReadLine();

    }
    tw.WriteLine("}");

    tr.Close();
    tw.Close();
}


//converts from a .txt or.bhv to an excel spreadsheet in basic format
//this format has the information pertinent to the common user
void convAdvXls(string ReadFileName, string SaveFileName)
{
    Excel.Application xlApp;
    Excel.Workbook xlWorkBook;
    Excel.Worksheet xlWorkSheet;
    object misValue = System.Reflection.Missing.Value;

    xlApp = new Excel.ApplicationClass();
    xlWorkBook = xlApp.Workbooks.Add(misValue);

    int i = 1;

    //creates the headers for the spreadsheet
    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);
    xlWorkSheet.Cells[i, 1] = "Model List";
    i++;
    xlWorkSheet.Cells[i, 1]  = "";
    xlWorkSheet.Cells[i, 2]  = "Name";
    xlWorkSheet.Cells[i, 3]  = "Classification";
    xlWorkSheet.Cells[i, 4]  = "Type";
    xlWorkSheet.Cells[i, 5]  = "Appearance";
    xlWorkSheet.Cells[i, 6]  = "ID";
    xlWorkSheet.Cells[i, 7]  = "Description";
    xlWorkSheet.Cells[i, 8]  = "Model FileName";
    xlWorkSheet.Cells[i, 9]  = "DIS Enumeration";
    xlWorkSheet.Cells[i, 10] = "EntityId";
    xlWorkSheet.Cells[i, 11] = "DeadReckoning";
    xlWorkSheet.Cells[i, 12] = "SmoothPosition";
    xlWorkSheet.Cells[i, 13] = "SmoothOrientation";
    xlWorkSheet.Cells[i, 14] = "MachineGunRoundsToDestroy";
    xlWorkSheet.Cells[i, 15] = "Accessories";
    xlWorkSheet.Cells[i, 16] = "IdleAnimations";
    xlWorkSheet.Cells[i, 17] = "MoveAnimations";
    i++;
```

```
bool keepgoing = true;
int count = 1;
string readline;
System.IO.TextReader tr = new System.IO.StreamReader(ReadFileName);

while (keepgoing)
{
   while (true)
   {
      readline = tr.ReadLine();

      if (readline == null)
      {
         keepgoing = false;
         break;
      }

      if (readline.Trim() == "behavior {" || readline.Trim() == "behavior{")
      {
         break;
      }

   }

   if (!keepgoing)
   {
      break;
   }

   //number
   xlWorkSheet.Cells[i, 1] = count.ToString();

   //name
   readline = tr.ReadLine().Trim();
   findQuotes(readline);
   xlWorkSheet.Cells[i, 2] = readline.Substring(m_Quote1 + 1, m_Quote2 - m_Quote1 -
1);

   //classification
   readline = tr.ReadLine().Trim();
   findChevron(readline);
   xlWorkSheet.Cells[i, 3] = readline.Substring(m_Chevron + 1).Trim();

   //type
   readline = tr.ReadLine().Trim();
```

```
            findQuotes(readline);
            xlWorkSheet.Cells[i, 4] = readline.Substring(m_Quote1 + 1, m_Quote2 - m_Quote1 -
1);

            //appearance
            readline = tr.ReadLine().Trim();
            findQuotes(readline);
            xlWorkSheet.Cells[i, 5] = readline.Substring(m_Quote1 + 1, m_Quote2 - m_Quote1 -
1);

            //id
            readline = tr.ReadLine().Trim();
            findChevron(readline);
            xlWorkSheet.Cells[i, 6] = readline.Substring(m_Chevron + 1).Trim();

            //description(country of origin)
            readline = tr.ReadLine().Trim();
            findQuotes(readline);
            xlWorkSheet.Cells[i, 7] = readline.Substring(m_Quote1 + 1, m_Quote2 - m_Quote1 -
1).Trim();

            //modelfilename
            readline = tr.ReadLine().Trim();
            findQuotes(readline);
            xlWorkSheet.Cells[i, 8] = readline.Substring(m_Quote1 + 1, m_Quote2 - m_Quote1 -
1).Trim();

            //dis enumeration
            tr.ReadLine();
            readline = tr.ReadLine().Trim();
            findChevron(readline);
            xlWorkSheet.Cells[i, 9] = readline.Substring(m_Chevron + 1).Trim();

            //entity id
            readline = tr.ReadLine().Trim();
            findChevron(readline);
            xlWorkSheet.Cells[i, 10] = readline.Substring(m_Chevron + 1).Trim();

            //dead reckoning
            readline = tr.ReadLine().Trim();
            findChevron(readline);
            xlWorkSheet.Cells[i, 11] = readline.Substring(m_Chevron + 1).Trim();

            //smooth position
            readline = tr.ReadLine().Trim();
            findChevron(readline);
```

```
            xlWorkSheet.Cells[i, 12] = readline.Substring(m_Chevron + 1).Trim();

            //smooth orientation
            readline = tr.ReadLine().Trim();
            findChevron(readline);
            xlWorkSheet.Cells[i, 13] = readline.Substring(m_Chevron + 1).Trim();

            //accesories
            tr.ReadLine().Trim();
            readline = tr.ReadLine().Trim();
            if (readline.StartsWith("accessories"))
            {
               findQuotes(readline);
               xlWorkSheet.Cells[i, 15] = readline.Substring(m_Quote1 + 1, m_Quote2 -
m_Quote1 - 1).Trim();
               readline = tr.ReadLine().Trim();
            }
            //idle animations
            if (readline.StartsWith("idleAnimations"))
            {
               findQuotes(readline);
               xlWorkSheet.Cells[i, 16] = readline.Substring(m_Quote1 + 1, m_Quote2 -
m_Quote1 - 1).Trim();
               readline = tr.ReadLine().Trim();
            }
            //move animations
            if (readline.StartsWith("moveAnimations"))
            {
               findQuotes(readline);
               xlWorkSheet.Cells[i, 17] = readline.Substring(m_Quote1 + 1, m_Quote2 -
m_Quote1 - 1).Trim();
               readline = tr.ReadLine().Trim();
            }
            //mchine gun rounds to destroy
            if (readline.StartsWith("machineGunRoundsToDestroy"))
            {
               findChevron(readline);
               xlWorkSheet.Cells[i, 14] = readline.Substring(m_Chevron + 1).Trim();  //the 14 is
not a typo
               readline = tr.ReadLine().Trim();
            }

            count++;
            i++;
         }
         tr.Close();
```

```
            xlWorkBook.SaveAs(SaveFileName, Excel.XlFileFormat.xlWorkbookNormal,
misValue, misValue, misValue, misValue, Excel.XlSaveAsAccessMode.xlExclusive, misValue,
misValue, misValue, misValue, misValue);
            xlWorkBook.Close(true, misValue, misValue);
            xlApp.Quit();

        }

        //converts from a .txt or.bhv to an excel spreadsheet in advanced format
        //this format has the complete information contained in the text document
        void convSimpleXls(string ReadFileName, string SaveFileName)
        {
            Excel.Application xlApp;
            Excel.Workbook xlWorkBook;
            Excel.Worksheet xlWorkSheet;
            object misValue = System.Reflection.Missing.Value;

            xlApp = new Excel.ApplicationClass();
            xlWorkBook = xlApp.Workbooks.Add(misValue);

            int i = 1;

            //creates the headers for the spreadsheet
            xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);
            xlWorkSheet.Cells[i, 1] = "Model List";
            i++;
            xlWorkSheet.Cells[i, 1] = "";
            xlWorkSheet.Cells[i, 2] = "Name";
            xlWorkSheet.Cells[i, 3] = "Classification";
            xlWorkSheet.Cells[i, 4] = "ID";
            xlWorkSheet.Cells[i, 5] = "Description";
            xlWorkSheet.Cells[i, 6] = "Model FileName";
            xlWorkSheet.Cells[i, 7] = "DIS Enumeration";
            i++;

            bool keepgoing = true;
            int count = 1;
            string readline;
            System.IO.TextReader tr = new System.IO.StreamReader(ReadFileName);


            while (keepgoing)
            {
                while (true)
                {
```

```
            readline = tr.ReadLine();

            if (readline == null)
            {
               keepgoing = false;
               break;
            }

            if (readline.Trim() == "behavior {" || readline.Trim() == "behavior{")
            {
               break;
            }

         }

         if (!keepgoing)
         {
            break;
         }

         //number
         xlWorkSheet.Cells[i, 1] = count.ToString();

         //name
         readline = tr.ReadLine().Trim();
         findQuotes(readline);
         xlWorkSheet.Cells[i, 2] = readline.Substring(m_Quote1 + 1, m_Quote2 - m_Quote1 -
1);

         //classification
         readline = tr.ReadLine().Trim();
         findChevron(readline);
         xlWorkSheet.Cells[i, 3] = readline.Substring(m_Chevron + 1).Trim();

         //id
         tr.ReadLine();
         tr.ReadLine();
         readline = tr.ReadLine().Trim();
         findChevron(readline);
         xlWorkSheet.Cells[i, 4] = readline.Substring(m_Chevron + 1).Trim();

         //description(country of origin)
         readline = tr.ReadLine().Trim();
         findQuotes(readline);
         xlWorkSheet.Cells[i, 5] = readline.Substring(m_Quote1 + 1, m_Quote2 - m_Quote1 -
1).Trim();
```

```
            //modelfilename
            readline = tr.ReadLine().Trim();
            findQuotes(readline);
            xlWorkSheet.Cells[i, 6] = readline.Substring(m_Quote1 + 1, m_Quote2 - m_Quote1 -
1).Trim();

            //dis enumeration
            tr.ReadLine();
            readline = tr.ReadLine().Trim();
            findChevron(readline);
            xlWorkSheet.Cells[i, 7] = readline.Substring(m_Chevron + 1).Trim();

            count++;
            i++;
         }
         tr.Close();

         object format = null;
         if(SaveFileName.EndsWith(".xls"))
         {
            format = Excel.XlFileFormat.xlWorkbookNormal;
         }
         else if(SaveFileName.EndsWith(".xlsx"))
         {
            format = Excel.XlFileFormat.xlWorkbookNormal; //todo save as .xlsx format
         }


         xlWorkBook.SaveAs(SaveFileName, format, misValue, misValue, misValue, misValue,
Excel.XlSaveAsAccessMode.xlExclusive, misValue, misValue, misValue, misValue, misValue);
         xlWorkBook.Close(true, misValue, misValue);
         xlApp.Quit();

      }


      void convertSimpleDelim(string ReadFileName, string SaveFileName, string d)
      {
         bool keepgoing = true;
         int count = 1;
         string readline;
         string[] writeline = new string[6];
         System.IO.TextReader tr = new System.IO.StreamReader(ReadFileName);
         System.IO.TextWriter tw = new System.IO.StreamWriter(SaveFileName);
```

```
tw.WriteLine(d + "Model List");
tw.WriteLine(d + "Name" + d + "Classification" + d +  "ID" + d +
   "Description" + d + "Model FileName" + d + "DIS Enumeration");

while (keepgoing)
{
   while (true)
   {
      readline = tr.ReadLine();

      if (readline == null)
      {
         keepgoing = false;
         break;
      }

      if (readline.Trim() == "behavior {" || readline.Trim() == "behavior{")
      {
         break;
      }

   }

   if (!keepgoing)
   {
      break;
   }

   //name
   readline = tr.ReadLine().Trim();
   findQuotes(readline);
   writeline[0] = readline.Substring(m_Quote1 + 1, m_Quote2 - m_Quote1 - 1);


   //classification
   readline = tr.ReadLine().Trim();
   findChevron(readline);
   writeline[1] = readline.Substring(m_Chevron + 1).Trim();

   //id
   tr.ReadLine();
   tr.ReadLine();
   readline = tr.ReadLine().Trim();
   findChevron(readline);
   writeline[2] = readline.Substring(m_Chevron + 1).Trim();
```

```
            //description(country of origin)
            readline = tr.ReadLine().Trim();
            findQuotes(readline);
            writeline[3] = readline.Substring(m_Quote1 + 1, m_Quote2 - m_Quote1 - 1).Trim();

            //modelfilename
            readline = tr.ReadLine().Trim();
            findQuotes(readline);
            writeline[4] = readline.Substring(m_Quote1 + 1, m_Quote2 - m_Quote1 - 1).Trim();

            //dis enumeration
            tr.ReadLine();
            readline = tr.ReadLine().Trim();
            findChevron(readline);
            writeline[5] = readline.Substring(m_Chevron + 1).Trim();

            tw.WriteLine("{0}" + d + "{1}" + d + "{2}" + d + "{3}" + d + "{4}" + d + "{5}"+ d +
"{6}",
                count, writeline[0], writeline[1], writeline[2], writeline[3], writeline[4],
writeline[5]);
            count++;
         }
         tr.Close();
         tw.Close();
      }

      void convertAdvDelim(string ReadFileName, string SaveFileName, string d)
      {
         bool keepgoing = true;
         int count = 1;
         string readline;
         string[] writeline;
         System.IO.TextReader tr = new System.IO.StreamReader(ReadFileName);
         System.IO.TextWriter tw = new System.IO.StreamWriter(SaveFileName);

         tw.WriteLine(d + "Model List");
         tw.WriteLine(d + "Name" + d + "Classification" + d + "Type" + d + "Appearance" + d +
"ID" + d +
            "Description" + d + "Model FileName" + d + "DIS Enumeration" + d + "EntityId" + d
+
            "DeadReckoning" + d + "SmoothPosition" + d + "SmoothOrientation" + d +
"MachineGunRoundsToDestroy"
            + d + "Accessories" + d + "IdleAnimations" + d + "MoveAnimations");

         while (keepgoing)
         {
```

```
while (true)
{
   readline = tr.ReadLine();

   if (readline == null)
   {
      keepgoing = false;
      break;
   }

   if (readline.Trim() == "behavior {" || readline.Trim() == "behavior{")
   {
      break;
   }

}

if (!keepgoing)
{
   break;
}
writeline = null;
writeline = new String[16];
//name
readline = tr.ReadLine().Trim();
findQuotes(readline);
writeline[0] = readline.Substring(m_Quote1 + 1, m_Quote2 - m_Quote1 - 1);

//classification
readline = tr.ReadLine().Trim();
findChevron(readline);
writeline[1] = readline.Substring(m_Chevron + 1).Trim();

//type
readline = tr.ReadLine().Trim();
findQuotes(readline);
writeline[2] = readline.Substring(m_Quote1 + 1, m_Quote2 - m_Quote1 - 1);

//appearance
readline = tr.ReadLine().Trim();
findQuotes(readline);
writeline[3] = readline.Substring(m_Quote1 + 1, m_Quote2 - m_Quote1 - 1);

//id
readline = tr.ReadLine().Trim();
findChevron(readline);
```

```
            writeline[4] = readline.Substring(m_Chevron + 1).Trim();

            //description(country of origin)
            readline = tr.ReadLine().Trim();
            findQuotes(readline);
            writeline[5] = readline.Substring(m_Quote1 + 1, m_Quote2 - m_Quote1 - 1).Trim();

            //modelfilename
            readline = tr.ReadLine().Trim();
            findQuotes(readline);
            writeline[6] = readline.Substring(m_Quote1 + 1, m_Quote2 - m_Quote1 - 1).Trim();

            //dis enumeration
            tr.ReadLine();
            readline = tr.ReadLine().Trim();
            findChevron(readline);
            writeline[7] = readline.Substring(m_Chevron + 1).Trim();

            //entity id
            readline = tr.ReadLine().Trim();
            findChevron(readline);
            writeline[8] = readline.Substring(m_Chevron + 1).Trim();

            //dead reckoning
            readline = tr.ReadLine().Trim();
            findChevron(readline);
            writeline[9] = readline.Substring(m_Chevron + 1).Trim();

            //smooth position
            readline = tr.ReadLine().Trim();
            findChevron(readline);
            writeline[10] = readline.Substring(m_Chevron + 1).Trim();

            //smooth orientation
            readline = tr.ReadLine().Trim();
            findChevron(readline);
            writeline[11] = readline.Substring(m_Chevron + 1).Trim();

            //accesories
            tr.ReadLine().Trim();
            readline = tr.ReadLine().Trim();
            if (readline.StartsWith("accessories"))
            {
               findQuotes(readline);
               writeline[13] = readline.Substring(m_Quote1 + 1, m_Quote2 - m_Quote1 -
1).Trim();
```

```
            readline = tr.ReadLine().Trim();
          }
          //idle animations
          if (readline.StartsWith("idleAnimations"))
          {
            findQuotes(readline);
            writeline[14] = readline.Substring(m_Quote1 + 1, m_Quote2 - m_Quote1 -
1).Trim();
            readline = tr.ReadLine().Trim();
          }
          //move animations
          if (readline.StartsWith("moveAnimations"))
          {
            findQuotes(readline);
            writeline[15] = readline.Substring(m_Quote1 + 1, m_Quote2 - m_Quote1 -
1).Trim();
            readline = tr.ReadLine().Trim();
          }
          //mchine gun rounds to destroy
          if (readline.StartsWith("machineGunRoundsToDestroy"))
          {
            findChevron(readline);
            writeline[12] = readline.Substring(m_Chevron + 1).Trim();  //the 14 is not a typo
            readline = tr.ReadLine().Trim();
          }

          tw.WriteLine("{0}" + d + "{1}" + d + "{2}" + d + "{3}" + d + "{4}" + d + "{5}"
            + d + "{6}" + d + "{7}" + d + "{8}" + d + "{9}" + d + "{10}" + d + "{11}" + d +
"{12}"
            + d + "{13}" + d + "{14}" + d + "{15}" + d + "{16}",
            count, writeline[0], writeline[1], writeline[2], writeline[3], writeline[4], writeline[5],
            writeline[6], writeline[7], writeline[8], writeline[9], writeline[10], writeline[11],
            writeline[12], writeline[13], writeline[14], writeline[15]);
          count++;
        }
        tr.Close();
        tw.Close();
      }
    }
}
```

INITIAL DISTRIBUTION LIST

		Copies
Weapon Systems Technology Information Analysis Center Alion Science and Technology 201 Mill Street Rome, NY 13440	Ms. Gina Nash gnash@alionscience.com	Electronic
Defense Technical Information Center 8725 John J. Kingman Rd., Suite 0944 Fort Belvoir, VA 22060-6218	Mr. Jack L. Rike jrike@dtic.mil	Electronic
AMSAM-LI	Ms. Anne C. Lanteigne anne.lanteigne@us.army.mil	Electronic
RDMR		
RDMR-CSI		
RDMR-WDG-C	Mr. Milton A. Lamb milton.lamb@us.army.mil	Electronic/Hardcopy